Utilizing Soft Information in Decoding of Variable Length Codes

Jiangtao Wen[1] and John D. Villasenor[2]

[1]M4 Labs
3550 General Atomics Court
Building 14, Suite 140
San Diego, CA 92121
gwen@packetvideo.com

[2]Electrical Engineering Department
University of California, Los Angeles
405 Hilgard Avenue
Los Angeles, CA 90024-1594
villa@icsl.ucla.edu

Abstract: *We present a method for utilizing soft information in decoding of variable length codes (VLCs). When compared with traditional VLC decoding, which is performed using "hard" input bits and a state machine, the soft-input VLC decoding offers improved performance in terms of packet and symbol error rates. Soft-input VLC decoding is free from the risk, encountered in hard decision VLC decoders in noisy environments, of terminating the decoding in an unsynchronized state, and it offers the possibility to exploit a priori knowledge, if available, of the number of symbols contained in the packet.*

1 Introduction

In most applications of variable length codes (VLCs), decoding is performed bit by bit, with the input to the entropy decoder assumed to be a sequence of "hard" bits about which no soft information is available. However, in noisy environments, soft information can be associated with each information bit, either by direct use of channel observations in the case of uncoded transmission, or through soft-output channel decoders when channel coding is used. It is intuitive that this soft information, if it can be exploited, can be used to improve the performance of VLC decoding.

The main contribution of the present paper is the introduction of a soft-in soft-out (SISO) dynamic decoding algorithm for decoding of VLCs. The SISO approach is inspired by SISO channel decoding algorithms such as the soft-output Viterbi algorithm (SOVA) [1] and methods for decoding of turbo codes [2], although the application to VLCs is quite different in concept and implementation. In contrast to the joint source/channel coding algorithm for VLCs in [3], the SISO VLC decoder proposed in this paper involves *no* modification to the encoder, it simply receives as input a packet of known length containing VLC data that has been corrupted by additive white Gaussian noise (AWGN), and produces the codeword sequence which is most likely to have been input to the VLC encoder at the transmitter. Simulation results show significant improvement in performance relative to traditional hard decision decoding of VLCs. The SISO method performs best when *a priori* information regarding the number of symbols represented in the packet is available, though even in the case the number of symbols is unknown, the performance is superior to hard decision methods.

2 SISO algorithm description

We consider a source producing symbols selected from an alphabet $\mathbf{X} = \{X_1, X_2, \cdots X_K\}$, where symbol X_k occurs with probability p_k and is represented using a binary codeword x_k of l_k bits. The alphabet size K may be infinite. The output of the VLC encoder is transmitted using packets of length L bits, representing N symbols. We use the integer $c(i)$ to denote the index to the symbol in the i-th position in the packet; for example, if $c(i) = j$, then the symbol in the ith position is X_j. The vector of symbol indices $\boldsymbol{C} = \{c(1), c(2), \cdots c(N)\}$ is mapped by the VLC encoding to the binary sequence represented by the concatenation $\{x_{c(1)} x_{c(2)} \cdots x_{c(N)}\}$.

The information available at the input to the VLC decoder is the observation vector containing L noise-corrupted bits $\mathbf{O} = \{o_1, o_2, \cdots, o_j, \cdots, o_L\}$ that can be divided into N subvectors $\{\boldsymbol{O}_i\}_{i=1}^{N}$, each containing the $l_{c(i)}$ bits representing the symbol $X_{c(i)}$, so that $\mathbf{O}$ can also be written as the concatenation $\mathbf{O} = \{\mathbf{O}_1 \mathbf{O}_2 \cdots \mathbf{O}_N\}$. Based on the observation vector $\boldsymbol{O}$, the optimal decoder selects the symbol vector $\boldsymbol{C}$ that maximizes $Prob\{\boldsymbol{C}|\boldsymbol{O}\}$, the posterior probability of the symbol sequence given the observation. This is equivalent to maximizing $Prob\{\boldsymbol{C}, \boldsymbol{O}\}$, the joint probability of the symbol sequence and the observation. The vector that maximizes $Prob\{\mathbf{C}, \mathbf{O}\}$ is denoted as $\mathbf{C}^* = \{c^*(1), c^*(2), \cdots c^*(N)\}$, and is associated with probability $P^* = Prob\{\mathbf{C}^*, \mathbf{O}\} = \max_{\mathbf{C}} Prob\{\mathbf{C}, \mathbf{O}\}$. For a packet of L bits and N symbols, $\boldsymbol{C}^*$ and P^* are calculated from Bellman's principle of dynamic programming [4], which leads to recursive equations as follows:

$$\begin{aligned} \boldsymbol{C}^*(\mathrm{N},\mathrm{L}) &= \boldsymbol{C}^*(N-1, L-l_{c^*(N)}) \cup c^*(N), \\ P^*(N,L) &= P^*(N-1, L-l_{c^*(N)}) \cdot Prob\{o_{L-l_{c^*(N)}+1}, \cdots, o_L | c(N) = c^*(N)\} \cdot p_{c^*(N)}, \end{aligned} \tag{1}$$

with

$$\begin{aligned} \mathrm{c}^*(\mathrm{N}) &= \arg \max_{i, i \in [1,K]} \{P^*(N-1, L-l_i) \cdot Prob\{o_{L-l_i+1}, o_{L-l_i+2}, \cdots, o_L | c(N) = i\} \cdot p_i\}, \\ C^*(1,L) &= \arg \max_{i, i \in [1,K]} Prob\{o_1, o_2, \cdots, o_L | c(1) = i\} \cdot p_i. \end{aligned} \tag{2}$$

The principal underlying equation (1) is that if the optimal last codeword in the packet is a codeword of length $l_{c^*(N)}$, the rest of the optimal packet, corresponding to $N-1$ symbols, must be the optimal $N-1$ symbol packet corresponding to the first $L - l_{c^*(N)}$ received values. Equations (2) and (2) involve selecting from the alphabet the symbol that is most likely, as measured by the products of probabilities that constitute the argument for the max function. The "SISO" terminology is used to describe the algorithm because the decoder provides not only the optimal sequence $\boldsymbol{C}^*$ but also the probability, or confidence level P^* associated with that sequence.

As an example, consider a simple VLC with an alphabet of size $K = 3$, with symbols X_1, X_2, and X_3 and corresponding codewords x_1=0, x_2= 10, and x_3=11. We assume that the probabilities of symbols are ideally matched to the code lengths l_i; e.g. that $p_i = 2^{-l_i}$. In transmitting the encoded sequence over the AWGN channel with noise standard deviation σ, we represent binary 0 by -1, and binary 1 by +1. Consider a 3-bit packet "010" correspoding to the symbol vector $\boldsymbol{C} = \{c(1) = 1, c(2) = 2\}$ and transmitted as $\{-1, +1, -1\}$ and received as $\boldsymbol{O} = \{-0.8, -0.05, -0.2\}$. Because the noise (and thus the observed value for each bit) is a continuous random variable, the SISO algorithm should

optimize the joint probability density of packet makeup and observation, evaluated at the received observation vector, rather than the joint probability. The maximization of equation (2) becomes minimization of the cumulative square error between the received and the originally transmitted values. Let $D^*(i,j)$ denote the global minimal cumulative square distortion of the first j bits of the packet containing exactly i codewords. Equations (1) and (2) give $\boldsymbol{C}^*(2,3) = \boldsymbol{C}^*(1,3-l_{c^*(2)}) \cup c^*(2)$, and

$$\begin{aligned} \mathrm{c}^*(2) &= \arg\min_{k=1,2,3} \left\{ D^*(1,3-l_k) + (o_{4-l_k} - x_{k1})^2 \cdots + (o_3 - x_{kl_k})^2 \right\} \\ &= \arg\min \Big\{ D^*(1,1) + (o_2 - x_{21})^2 + (o_3 - x_{22})^2, \\ &\qquad D^*(1,2) + (o_3 - x_{11})^2, D^*(1,1) + (o_2 - x_{31})^2 + (o_3 - x_{32})^2 \Big\}, \quad (3) \end{aligned}$$

Given the observation $\mathbf{O} = \{-0.8, -0.05, -0.2\}$, equation (3) involves finding the smaller of $D^*(1,1) + \min\{[(-0.05-1)^2 + (-0.2+1)^2], [(-0.05-1)^2 + (-0.2-1)^2]\} = D^*(1,1) + 1.7425$, and $D^*(1,2) + (-0.2+1)^2 = D^*(1,2) + 0.64$. Next, $D^*(1,1)$ and $D^*(1,2)$ are calculated as $D^*(1,1) = (-0.8+1.0)^2 = 0.04$, and

$$D^*(1,2) = \min\left\{ \left[(-0.8-1)^2 + (-0.05+1)^2\right], \left[(-0.8-1)^2 + (-0.05-1)^2\right] \right\} = 4.1425,$$

so we have $D^*(2,3) = \min\{D^*(1,1) + 1.7425, D^*(1,2) + 0.64\} = D^*(1,1) + 1.7425 = 1.7825$. Therefore, the first codeword is one bit in length and is most likely x_1, and the second codeword, which contains two bits, is most likely x_2 by the result of the first minimization involving $D^*(1,1)$ above, and the corresponding decoded symbol sequence is correctly identified as $\boldsymbol{C}^*(2,3) = \{c^*(1) = 1, c^*(2) = 2\}$. In contrast with the above, when the traditional bitwise hard-decision and look-up-table based decoding is used, the hard decisions made on the received values would be $\{-1,-1,-1\}$ which would be incorrectly decoded as $\{c^*(1) = 1, c^*(2) = 1, c^*(3) = 1\}$, the codeword x_1 repeated 3 times. The probability density P^* associated with output of the SISO decoder is the product of the codeword probabilities $p(1)p(2)$, the noise σ, and the metric $D^*(2,3)$:

$$P^* = p(1)p(2)\left(\sqrt{2\pi\sigma^2}\right)^{-3/2} exp(-D^*(2,3)/2\sigma^2).$$

Evaluating this using $D^*(2,3) = 1.7825$ and the appropriate value of σ would give the probability density associated with the decoded sequence. Note that in this example it was not necessary to use σ in carrying out the decoding, so σ affects only the probability density evaluated with the decoded sequence, not the decoded sequence itself. The numerical value of P^* is quite small in this example. For example, if $\sigma = 1$, $P^* = .0033$. This occurs because the second bit of the sequence, which was transmitted as value $+1$ was received as $-.05$, corresponding to a large, and therefore improbable noise realization. In the absence of any *a priori* information regarding the number of symbols N in the packet, the most probable symbol sequence is indeed $\{c^*(1) = 1, c^*(2) = 1, c^*(3) = 1\}$. However, when the constraint that $N = 2$ is imposed, the most probable sequence becomes $\{c^*(1) = 1, c^*(2) = 2\}$.

Table 1 shows simulation results for packets containing $N = 100$ symbols drawn from the code x_1=0, x_2= 10, and x_3=11 using symbol probabilities $\{p_1, p_2, p_3\} = \{0.5, 0.49, 0.01\}$. For each value of noise standard deviation σ, 20 packets were produced using symbols following this probability distribution, and each packet was corrupted by noise and then VLC decoded. 1000 different noise realizations were used for each packet. Packet error rate gives the fraction of packets in which one or more errors are present in the output of

the VLC decoder. Symbol error rate gives the fraction of symbols which are incorrect. It is clear from the table that when the number of symbols N is known *a priori*, the SISO approach gives significantly better performance than hard decision decoding both in terms of packet and symbol error rates over a wide range of noise levels.

Table 1: Comparison of SISO and bit-wise hard decision (HD) based decoding

σ (SNR)	SISO decoding				HD decoding	
	Packet error rate		Symbol error rate		Packet error rate	Symbol error rate
	N known	N unknown	N known	N unknown		
0.3 (10 dB)	8.2×10^{-3}	4.5×10^{-2}	3.6×10^{-4}	1.0×10^{-2}	7.7×10^{-2}	1.0×10^{-2}
0.4 (8 dB)	0.23	0.43	0.028	0.10	0.64	0.11
0.5 (6 dB)	0.73	0.86	0.13	0.23	0.97	0.26

The need to know N can be a disadvantage in that this information will not always be present. Because the SISO decoder outputs "soft" confidence level (P^*) information associated with the optimal decoded sequence given N, we can use this output in the decoding. Table 1, also gives results for the case where the decoder does not know that there are $N = 100$ symbols in the packet, and explores all possible values of N between 70 and 130. This is a sufficient range given the average codeword length and the number of bits L in the packet.

As the example given previously illustrated, when the noise is strong, the most probable decoded sequence is not generally the one with the correct number of symbols. As expected, this leads to a penalty in performance for the SISO decoder as shown in the "N unknown" column in Table 1. The symbol error rate for SISO decoding when N is not known is only marginally superior to hard decision decoding, though the packet error rate still shows significant improvements, particularly as the SNR is increased.

Another advantage of SISO decoding is that it avoids the synchronization problem inherent to hard decision VLC decoders, which do not guarantee that the final bit of a packet will coincide with the final bit of a codeword. Furthermore, in hard decision VLC decoding, correct synchronization status at the end of the packet can occur even when there are decoding errors, due to the well known self resynchronizing property of VLC codes. Another error that can occur in hard decision VLC decoders is an output containing an incorrect number of symbols. Hard decision VLC decoders are highly effective at detecting these errors, but they can not easily correct them. The SISO decoder, while not strictly able to perform error correction, can at least impose a constraint on the number of symbols produced during decoding, and gives the output sequence that is most probable given the observation and subject to the constraint. These issues are explored in Table 2, which was generated using the same simulation conditions as Table 1.

Table 2: Statistics of invalid decoding results for hard decision (HD) decoding

σ (SNR)	HD decoder out-of-sync	HD outputs incorrect number of symbols		
		Probability of occurance	SISO performance for corresponding packets	
			Packet error rate	Symbol error rate
0.3 (10 dB)	5.0×10^{-5}	3.6×10^{-2}	2.8×10^{-3}	2.8×10^{-5}
0.4 (8 dB)	2.7×10^{-3}	0.31	0.24	0.029
0.5 (6 dB)	1.1×10^{-2}	0.69	0.73	0.13

The first column in Table 2 gives the fraction of packets for which hard decision decoding

terminates in an unsynchronized state. Considering these numbers along with the hard decision symbol error rate results from Table 1 confirms that even in the presence of large error rates, correct synchronization at the end of the packet can be highly probable in hard decision decoding. The second column in the table gives the fraction of packets for which the hard decision decoder terminates in a synchronized state but outputs an incorrect number of symbols. The third and fourth columns in the table explore the performance of the SISO decoder on these packets. This sheds light on the performance of the SISO decoder on packets which would lead to (typically uncorrectable) error indications by a hard decision decoders. The improvement shown by SISO decoding is significant. For example, for an SNR of 8 dB, the table shows that for hard decision decoding, 31% of the packets would be in error due to an incorrect number of symbols. In SISO decoding, the packet error rate is only 24%, meaning that 76% of the packets would be decoded with no error at all (correct number of symbols and all symbols correct). The symbol error rate is only 2.9%.

In the simulations above it is assumed that the distribution of symbol probabilities is correctly known at the decoder. Table 3 explores the case where the decoder has an incorrect estimate of the symbol probabilities. As before, the results in the table are generated using 1000 noise realizations applied to each of 20 different packets of size N=100 using the 3-symbol alphabet. The true symbol probabilities are $\{p_1, p_2, p_3\} = \{0.5, 0.49, 0.01\}$, and the table gives the performance when the decoder assumes symbol probabilities of $\{p_1, p_2, p_3\} = \{0.5, 0.25, 0.25\}$ and $\{p_1, p_2, p_3\} = \{0.5, 0.4999, 0.0001\}$. Comparison with the SISO decoding results in Table 1 shows that penalty for mismatch is larger in relative terms for the low noise case, though the absolute error rates remain low despite the mismatch. Even when the decoder uses incorrect probability estimates, it can still utilize information on synchronization and the number of codewords N in the decoding process.

Table 3: SISO decoding with unmatched codeword probabilities

σ (SNR)	Assumed probabilities at the decoder			
	$\{0.5, 0.25, 0.25\}$		$\{0.5, 0.4999, 0.0001\}$	
	Packet error rate	Symbol error rate	Packet error rate	Symbol error rate
0.3 (10 dB)	3.6×10^{-2}	7.6×10^{-4}	3.0×10^{-2}	6.6×10^{-4}
0.4 (8 dB)	0.46	0.032	0.37	0.030
0.5 (6 dB)	0.93	0.14	0.83	0.13

3 Source distribution estimation

The loss in performance due to lack of knowledge of the source probability distribution raises the issue of whether the distribution can be estimated as part of the decoding process. Dynamic source distribution estimation avoids the need to transmit explicit probability information as side information, and also enables the decoder to track the probabilities when the distribution is nonstationary.

VLC encoding followed by transmission through a noisy channel can be modeled based on a doubly stochastic process very similar to a conventional hidden Markov model (HMM) ([5],[6]). Techniques such as the Baum-Welch algorithm for estimating HMM parameters are well known in the literature ([6], [7]) though they can not be directly applied in the case of VLCs because the codewords are of variable length, and the received packet can be parsed in multiple ways. We introduce a generalized forward-backward algorithm that addresses

this problem. Figure 1 illustrates the HMM for a VLC with three codewords x_1=0, x_2= 10, and x_3=11. While in the examples in the previous sections using this VLC, the source was i.i.d., the algorithm described below works for sources with memory as well. Each state is designated by an integer which is the index to the most recent codeword generated by the source. Branches are labeled with the transition probability and the codeword produced when the branch is traversed. The i.i.d. source is the special case where all probabilities entering each state are equal.

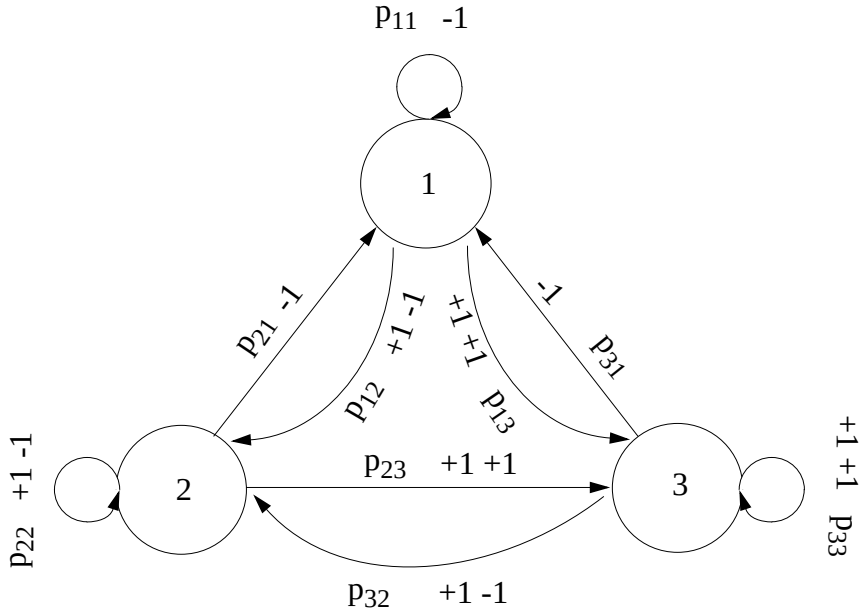

Figure 1: A Doubly Stochastic Model for VLC Encoding. p_{ij} designates the transition probability from state i to state j. Values on the branches correspond to the binary codeword produced when the corresponding transition is made.

Denoting the position of the last bit of the t-th codeword in the N-symbol L-bit packet by the random variable L_t, we can define a forward recursion metric $\alpha_t(i,j)$ that expresses the joint probability of the observation through the t-th codeword, of the state c_t (here we use c_t to denote the state at discrete time t because from states in the model of Figure 1 are designated by the index to the latest codeword transmitted) after transmission of the t-th codeword, and of L_t taking on value j:

$$\alpha_{\rm t}({\rm i,j}) = {\rm Prob}\{{\rm O_1, O_2, \cdots, O_t, c_t = i, L_t = j}\}, \tag{4}$$

Recursive computation of α proceeds using

$$\begin{cases} \alpha_1(i,j) = \pi(i)\delta(l_i,j)b(i,j,O), \\ \alpha_{t+1}(i,j) = [\sum_k \alpha_t(k, j-l_i) Prob\{c_{t+1}=i | c_t = k, L_t = j-l_i\}]\, b(i,j,O), \quad \forall 1 \le t \le N-1, \end{cases} \tag{5}$$

where $\delta(i,j)$ is the Kroneker delta function, $b(i,j,O) = Prob(o_{j-l_i+1}, o_{j-l_i+2}, \cdots, o_j | x_i)$, $\pi(i)$ is the probability that the first codeword of the packet is x_i, and $Prob\{c_{t+1} = i | c_t = k, L_t = j\}$ is the probability for the encoder to output codeword i given that the t-th codeword in the packet is k, and that the last bit of codeword k is the j-th bit of the packet. Because symbol x_i can be the $(t+1)$-th symbol in an L-bit, N-symbol packet when

$L_t = j$ if and only if both of the probabilities $P^*(t,j)$ and $P^*(N-t-1, L-j-l_i)$ are greater than zero, $Prob\{c_{t+1} = i | c_t = k, L_t = j\}$ can be derived from the transition matrix of the encoder by

$$\begin{aligned} & Prob\{c_{t+1} = i | c_t = k, L_t = j\} \\ = & \begin{cases} 0, & P^*(t,j) = 0 \text{ or } P^*(N-t-1, L-j-l_i) = 0 \\ \frac{p_{ki}}{\sum_{m, P^*(N-t-1, L-j-l_m)>0} p_{km}} & \text{otherwise.} \end{cases} \end{aligned} \tag{6}$$

The above formulation ensures that that only those state transitions that allow the possibility of producing a valid packet containing L bits and N symbols (and therefore have $P^*(t,j) > 0$ are considered.

Similar to the definition of α, we can define $\beta_t(i,j) = Prob\{O_{t+1}, O_{t+2}, \cdots O_N, L_t = j | c_t = i\}$, which is computed using the backward recursion

$$\begin{cases} \beta_N(i,j) = \delta(L,j) b(i,j,O), \\ \beta_t(i,j) = \sum_k \beta_{t+1}(k, j+l_k) Prob\{c_{t+1} = k | c_t = i, L_t = j\} b(k, j+l_k, O), \quad \forall t = N-1, \cdots, 1. \end{cases} \tag{7}$$

To update HMM parameters, we define

$$\begin{aligned} \gamma_t(\mathrm{i,j}) &= Prob\{c_t = i, L_t = j | O\} \\ &= \frac{\alpha_t(i,j)\beta_t(i,j)}{Prob\{O\}}, \end{aligned} \tag{8}$$

$$\begin{aligned} \xi_t(i,k,j) &= Prob\{c_t = i, c_{t+1} = k, L_t = j | O\}, \\ &= \frac{\sum_l \alpha_t(i,l) Prob\{c_{t+1} = i | c_t = k, L_t = j\} \beta_{t+1}(l + l_k, k) b(k, l + l_k, O)}{Prob\{O\}}, \end{aligned} \tag{9}$$

where $Prob\{O\} = \sum_i \alpha_T(i, L)$.

We can update the distribution of the first codeword to $\{\pi'\}$ and the probabilities for codewords to $\{p'_{ik}\}$, using:

$$\begin{cases} \pi'(i) = \gamma_1(i), \\ p'_{ik} = \frac{\sum_{t=1,\cdots T-1, j=1,\cdots,L, Prob\{c_{t+1}=m | c_t = i, L_t = j\}>0, \forall m} \xi_t(i,j,k)}{\sum_{t=1,\cdots T-1, j=1,\cdots,L, Prob\{c_{t+1}=m | c_t = i, L_t = j\}>0, \forall m} \gamma_t(i,j)}. \end{cases} \tag{10}$$

Table 4 provides an example of the results of source distribution estimation. The results in the table are based on one packet containing 100 symbols from the VLC x_1=0, x_2= 10, and x_3=11, transmitted over a AWGN channel with different noise powers. As in the tables in the previous section, the noise power is expressed both in terms of the noise standard deviation (where the transmitted signals are of unit power) and as an SNR. The strongest noise shown in the table, $\sigma = 2$, corresponds to a hard decision bit error rate of .31. The packet was created by drawing from a source with probabilities $\{p_1, p_2, p_3\} = \{0.5, 0.25, 0.25\}$, and the the actual code word relative frequencies appearing in the packet were $\{p_1, p_2, p_3\} = \{0.47, 0.26, 0.27\}$. The initial value used in the iteration was $\{p_1, p_2, p_3\} = \{0.8, 0.1, 0.1\}$. The table provides the probability estimates for $\{p_1, p_2, p_3\}$ for the case where the soft received values are used in the estimation process, and for the case where the values are subject to hard decisions prior to the input of the estimation algorithm. This allows reduction in complexity, and as the table shows, involves no significant loss in

accuracy. Of course, even if hard decisions are used for the estimation, the original soft observations need to be retained in memory so that the SISO VLC decoding algorithm described in the previous section can be utilized. Simulations with other codeword alphabets produced result consistent with Table 4 and confirm that the approach in equation (10) can produce an accurate estimate of the source distribution, even in the presence of a very low SNR.

Table 4: Source distribution estimation with forward-backward iteration

σ (SNR)	Estimated probabilities with/without hard decision		
	p_1	p_2	p_3
0.5 (6.0 dB)	0.480/0.480	0.275/0.275	0.245/0.245
1.0 (0.0 dB)	0.477/0.477	0.261/0.263	0.262/0.259
2.0 (-6.0 dB)	0.475/0.476	0.240/0.242	0.285/0.282
Actual Probabilities	0.470	0.260	0.270

We performed simulations in which the received data were first subject to the estimation process described above to establish codeword probabilities, and then input to the SISO decoder described in section 2. As before, packets containing 100 symbols were used, and no reduction in performance occured with respect to the SISO decoder in which the source distribution is known *a priori*.

4 Conclusions

We have presented an algorithm for using soft information in VLC decoding. Simulations demonstrate a significant improvement over traditional VLC decoding based on hard inputs. The method here exploits not only the presence of soft values, but also *a priori* knowledge of the number of bits L and, when available, symbols N. This also helps to improve the performance with respect to hard decision VLC methods, which offer no means to enforce consistency between the expected and actual number of symbols produced by the decoder. We have also proposed an efficient, generalized forward-backward reccursion that can be used to estimate source symbol distribution.

References

[1] M.P.C. Fossorier, F. Burkert, L. Shu, J. Hagenauer, "On the equivalence between SOVA and max-log-MAP decodings," *IEEE Comm. Letters*, vol.2, no.5, pp. 137-139, May, 1998.

[2] S. Benedetto, D. Divsalar, G. Montorsi, F. Pollara, "A soft-input soft-output APP module for iterative decoding of concatenated codes," *IEEE Comm. Letters,* vol. 1, no.1, pp. 22-24, Jan. 1997.

[3] N. Demir, K. Sayood, "Joint source/channel coding for variable length codes," *Proc. 1998 IEEE Data Compression Conference*, Snowbird, UT, March, 1998.

[4] R.E. Bellman, *Dynamic Programming and Modern Control*, Academic Press, 1966.

[5] L.R. Rabiner, B.H. Huang, "An introduction to hidden Markov Models," *IEEE ASSP Magazine*, pp. 4-16, Jan. 1986.

[6] L.E. Baum, J. Eagon, "An inequality with applications to statistical prediction for functions of Markov process and to a model for ecology," *Bull. American Math Society*, vol. 73, pp. 360-363, 1963.

[7] H. Lucke, "Which stochastic models allow Baum-Welch training?" *IEEE Trans. Signal Processing*, pp. 2746-2756, November 1996.

[8] L.R. Bahl, J. Cocke, F. Jelinek, J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. Information Theory*, pp. 284-287, March 1974.